International Journal of Mathematics and Computer Applications Research (IJMCAR)
ISSN(P): 2249-6955; ISSN(E): 2249-8060
Vol. 5, Issue 3, Jun 2015, 1-10
© TJPRC Pvt. Ltd.

IMPLEMENTATION AND PERFORMANCE ANALYSIS OF MODIFIED AES ALGORITHM WITH KEY-DEPENDENT DYNAMIC S-BOX AND KEY MULTIPLICATION

LAKSHMI R[1] & MOHAN H S[2]

[1]Research Scholar, Department of Information Science and Engineering, S. J. B. Institute of Technology, Bangalore, Karnataka, India

[2]Professor & Head of the Department, Department of Information Science and Engineering, S. J. B. Institute of Technology, Bangalore, Karnataka, India

ABSTRACT

In today's world, Internet has become the principal means of communication which has led to the increased use of computer communication system by almost all organizations. This has the risk of threat to the data flowing through the network. Encryption is the most extensively used cryptographic countermeasure for these threats. The most popular and strongest block cipher algorithm in symmetric key cryptography is Advanced Encryption Standard (AES). In this paper, performance comparison of Rijndael which is existing AES algorithm with modified AES algorithm through diffusion analysis in terms of First order Strict Avalanche Criteria (SAC) and Higher order SAC is done. This paper provides the use of dynamic S-Box which is dependent on the key provided by the user instead of standard S-Box which is used in the existing AES. The generated S-Boxes increase the complexity of the algorithm and cause the differential and linear cryptanalysis more difficult. The existing AES uses key addition round which is simpler and has less diffusion power. To overcome this, this paper also provides key multiplication method to enhance the diffusion power.

KEYWORDS: Advanced Encryption Standard, Strict Avalanche Criteria, Diffusion and Confusion, Differential and Linear Cryptanalysis

1. INTRODUCTION

Security is the principal goal in the design of cryptographic algorithm. The proliferation of computer usage and their interconnections through network have increased the need of protection of information stored against viruses, hackers, eavesdroppers, and electronic data deception.

The four basic functions provided by cryptographic algorithms are: 1) Authentication 2) Confidentiality 3) Non-repudiation and 4) Integrity. Mathematical functions like addition, transposition, substitution, rotation, etc., are used to randomize the cipher value. Such operations are repeated several iterations to achieve certain required diffusion level.

1.1. Cryptanalysis

There are two approaches to attack a conventional encryption algorithm.

Cryptanalysis: This is the type of attack which rely on the algorithm’s nature and some general or statistical property of the plaintext. The attacker may also have some sample plaintext-ciphertext pairs. This is the approach to deduce plaintext or key.

Brute-Force Attack: This is the type of attack where attacker tries every possible key on a sample ciphertext until an intelligible translation into plaintext is obtained. This type of attack is not possible with relatively lengthy key.

1.2. Confusion and Diffusion

These terms were introduced by Claude Shannon which define the basic techniques for any cryptographic systems to thwart cryptanalysis based on statistical analysis. Or in other words, all statistics of the ciphertext are independent of the key being used or the plaintext.

Diffusion is to make the statistical relationship between ciphertext and plaintext as complex as possible. That is, a small change in plaintext should make significant changes in ciphertext which is equivalent to saying that every single bit of ciphertext is affected by many plaintext bits. *Confusion* is to make the statistical relationship between ciphertext and key as complex as possible. That is a small change in key should make a significant changes in ciphertext. Every encryption algorithm does use the diffusion and confusion layers. The diffusion layer is based upon the linear operations such as multi-permutations, key additions, polynomial multiplication with some known constants etc. The confusion layer is based on the non-linear and complex operations such as Substitution Box (S-Box) (J. Daemen et al, 1999).

1.3. Strict Avalanche Criteria (SAC)

SAC is the desirable property of an encryption algorithm which states that a small change in plaintext or key should produce significant changes in ciphertext. There are two types of SAC: 1) First order SAC: It is expressed in terms of change in output bits when single bit is changed in input bits. 2) Higher order SAC: It is expressed in terms change in output bits when multiple bits are changed simultaneously in input bits.

2. ADVANCED ENCRYPTION STANDARD ALGORITHM

To replace the Data Encryption Standard (DES) algorithm, the National Institute of Standards and Technology (NIST) in 1997, issued a call for proposals to develop and choose a new AES. In 1998, NIST shortlisted fifteen candidate algorithms and later narrowed down to five algorithms based on security and efficiency characteristics. Finalists which are selected as AES candidate algorithms are MARS, RC6, Rijndael, Serpent, and Twofish. The performance comparison is available in (Mohan H.S et al, 2010). In November 2001, NIST selected Rijndael as the proposed AES algorithm due to its high level security, speed, flexibility and ease of implementation.

2.1. Rijndael Algorithm

It is a symmetric, block cipher algorithm developed by Joan Daemen and Vincent Rijmen. The algorithm supports key size of 128, 192, and 256 bits. It supports data block length of 128 bits only while conserving the property of three different key lengths. The 128 bit data block is organized as 4X4 matrix of bytes which is called the state. This state is fed to four basic block operations – ByteSub transformation, the ShiftRow transformation, the MixColumn transformation, and AddRoundKey.

- **ByteSub Transformation:** Non linear byte-by-byte substitution using S-Box, which is constructed by calculating multiplicative inverse followed by affine transformation.
- **ShiftRows Transformation:** Simple byte transposition which is a linear diffusion process operating on individual rows. The bytes in the last three rows of the state are cyclically shifted; the offset of the left shift varies from one to three bytes.
- **MixColumn Transformation:** It is equivalent to matrix multiplication over GF(2^8) of columns of the states. Each column vector is multiplied with a fixed matrix and the bytes are treated as polynomials rather than numbers.
- **AddRoundKey:** Simple XOR between the working state and the round key.

These operations are run in 10, 12, or 14 iterations depending on the size of the key – 128 bits, 192 bits, or 256 bits respectively. AES algorithm Encryption structure is depicted in Figure 1.

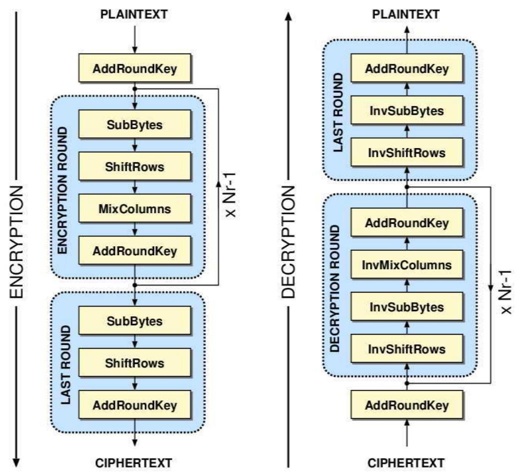


Figure 1: AES Algorithm Structure

All four operations described above have corresponding inverse operations such as Inv-SubBytes which uses inverse S-Box, Inv-ShiftRows, and Inv-MixColumn. Same sequence of operations but in the reverse order as one in the encryption structure is used as decryption structure. AddRoundKey step is its own inverse (Abd-ElGhafar et al, 2009).

In terms of the critical path between the ciphertext and plaintext, Rijndael is regarded as the fastest algorithm. In section 3, we introduce a new approach for the generation of S-Box which is dynamic in nature and dependent on the key provided. This results in the increased complexity of S-Box generation and hence the diffusion power. This makes the differential and linear cryptanalysis more difficult. In section 4, we introduce Key Multiplication operation instead of AddRoundKey which is much complex operation in terms of diffusion. In this paper we also make the performance comparison of Rijndael algorithm with modified AES with the above two modules mentioned in terms of CPU time and diffusion power analysis that is First order SAC and Higher order SAC.

3. GENERATION OF DYNAMIC S-BOX

S-Box is used to perform byte substitution operation. AES defines 16X16 matrix of byte values which contains a permutation of all possible 256values which are of 8-bit in size. Each individual byte of state is replaced by a new value from this S-Box in following way: The leftmost nibble is used as row index to the S-Box and rightmost nibble is used as column index to the S-Box. The corresponding inverse S-Box is used for decryption operation.

3.1. Related Studies

(Abd-ElGhafar et al, 2009) state that it is possible to generate our own S-Box by choosing different constant value which is used in the affine transformation. (J. Juremi et al, 2012) proposed a substitution box that makes use of the RC4 key schedule algorithm (KSA). The resulting matrix is a key-dependent S-box based that is dynamically generated. In (Mohan H. S. et al, 2011 (b)), authors have proposed another key dependent S-box that will substitute the Rijndael S-box. In their paper, they modified the AES cipher by placing another phase in the beginning of the round function. They call the extra phase as the S-box Rotation that will rearrange by way of rotating the Rijndael S-box according to a round key. The round key is derived from the cipher key using the key schedule algorithm. The rotation value is dependent on the entire round key. The results of their study showed that it does not violate the security of the cipher by doing the enhancement on the original AES. The enhanced version introduces confusion without violating the diffusion property.

3.2. Standard S-Box Construction

The standard S-Box is constructed in the following fashion:

- Initialize the S-Box with 1 to 256 values as 16X16 matrix in ascending order row by row.
- Find the multiplicative inverse of each byte in the S-Box in the finite GF(2^8)
- Apply affine transformation to each bit in each byte of S-Box in the following way. Consider that each bit in the byte of S-Box is labeled (b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0)

$$b' = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i$$

where c_i is the i-th bit of byte c with the value {63}, that is, c7c6c5c4c3c2c1c0) = (01100011). The prime (′) indicates that the variable is to be updated by the value on the right. The AES standard depicts this transformation in matrix form as follows:

$$\begin{bmatrix} b_0' \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Each element in the product matrix is the bitwise XOR of elements of one row and one column. Further, the final addition, shown in the above equation is a bitwise XOR, the inverse S-box is obtained by taking the inverse of equation, affine transformation followed by taking the multiplicative inverse in GF (2^8). As an example, consider the input value {95}. The multiplicative inverse in GF (2^8) is $\{95\}^{-1}$ = {8a}, which is 10001010 in binary. Using the above equation, the result is {2A}.

3.3. Proposed Modified AES with dynamic S-Box

The new dynamic S-Box is constructed based on the fact that the state always has permutation of all 8-bit numbers from 0 to 255. In this design, the key schedule algorithm is used to produce the permutation to generate multiple S-Boxes dynamically. This is constructed as the following steps:

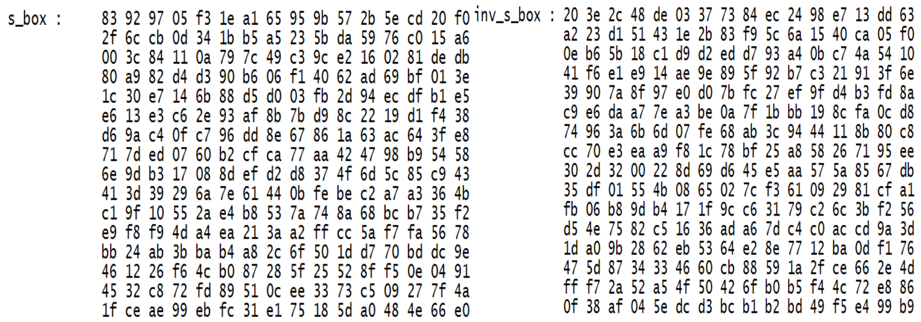
- The first S-Box is the Rijndael S-Box and is given as an array S which is constructed as explained in the section 3.2.
- After providing the input key, initialize as follows

```
J=0;
For i=0 to 255
k[i]=key[i mod keylength];
J=(j+s[i]+k[i])mod 256;
exchange (S[i], S[j])
```

- The output of the second step gives 256 different values and the generated permutation of values depend on the input key. If we change one byte value in the input key we get another permutation of 256 values. Hence it is possible to construct 256! S-Boxes depending on the input key.
- For the produced 256 values, apply affine transformation one again as applied in the original S-Box, to avoid any fixed points and to make new S-Box invertible.

Consider the S-Box generated for two sample keys. Keys are given in decimal number where 1 bye is expressed as 3 decimal digits.

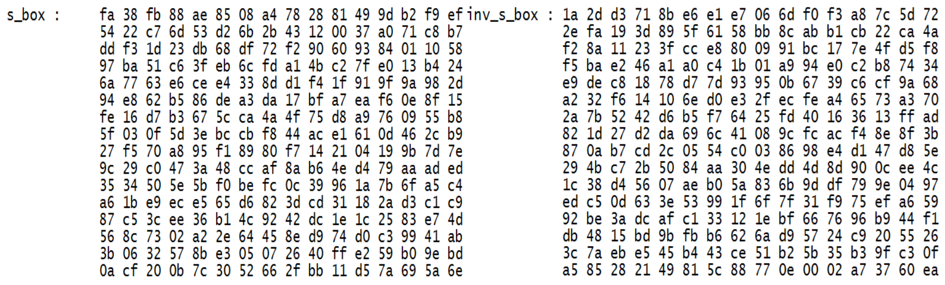
Example1: Key 1 =015025035045055065075085095105115125135145155165. The generated S-box is given in Figure 2



```
s_box :     83 92 97 05 f3 1e a1 65 95 9b 57 2b 5e cd 20 f0 inv_s_box : 20 3e 2c 48 de 03 37 73 84 ec 24 98 e7 13 dd 63
            2f 6c cb 0d 34 1b b5 a5 23 5b da 59 76 c0 15 a6             a2 23 d1 51 43 1e 2b 83 f9 5c 6a 15 40 ca 05 f0
            00 3c 84 11 0a 79 7c 49 c3 9c e2 16 02 81 de db             0e b6 5b 18 c1 d9 d2 ed d7 93 a4 0b c7 4a 54 10
            80 a9 82 d4 d3 90 b6 06 f1 40 62 ad 69 bf 01 3e             41 f6 e1 e9 14 ae 9e 89 5f 92 b7 c3 21 91 3f 6e
            1c 30 e7 14 6b 88 d5 d0 03 fb 2d 94 ec df b1 e5             39 90 7a 8f 97 e0 d0 7b fc 27 ef 9f d4 b3 fd 8a
            e6 13 e3 c6 2e 93 af 8b 7b d9 8c 22 19 d1 f4 38             c9 e6 da a7 7e a3 be 0a 7f 1b bb 19 8c fa 0c d8
            d6 9a c4 0f c7 96 dd 8e 67 86 1a 63 ac 64 3f e8             74 96 3a 6b 6d 07 fe 68 ab 3c 94 44 11 8b 80 c8
            71 7d ed 07 60 b2 cf ca 77 aa 42 47 98 b9 54 58             cc 70 e3 ea a9 f8 1c 78 bf 25 a8 58 26 71 95 ee
            6e 9d b3 17 08 8d ef d2 d8 37 4f 6d 5c 85 c9 43             30 2d 32 00 22 8d 69 d6 45 e5 aa 57 5a 85 67 db
            41 3d 39 29 6a 7e 61 44 0b fe be c2 a7 a3 36 4b             35 df 01 55 4b 08 65 02 7c f3 61 09 29 81 cf a1
            c1 9f 10 55 2a e4 b8 53 7a 74 8a 68 bc b7 35 f2             fb 06 b8 9d b4 17 1f 9c c6 31 79 c2 6c 3b f2 56
            e9 f8 f9 4d a4 ea 21 3a a2 ff cc 5a f7 fa 56 78             d5 4e 75 82 c5 16 36 ad a6 7d c4 c0 ac cd 9a 3d
            bb 24 ab 3b ba b4 a8 2c 6f 50 1d d7 70 bd dc 9e             1d a0 9b 28 62 eb 53 64 e2 8e 77 12 ba 0d f1 76
            46 12 26 f6 4c b0 87 28 5f 25 52 8f f5 0e 04 91             47 5d 87 34 33 46 60 cb 88 59 1a 2f ce 66 2e 4d
            45 32 c8 72 fd 89 51 0c ee 33 73 c5 09 27 7f 4a             ff f7 2a 52 a5 4f 50 42 6f b0 b5 f4 4c 72 e8 86
            1f ce ae 99 eb fc 31 e1 75 18 5d a0 48 4e 66 e0             0f 38 af 04 5e dc d3 bc b1 b2 bd 49 f5 e4 99 b9
```

Figure 2: S-Box Generated for Key 1

Example2: Key 2 =111121131141151161171181191112122132142152162172. The generated S-box is given in Figure 3



```
s_box :     fa 38 fb 88 ae 85 08 a4 78 28 81 49 9d b2 f9 ef inv_s_box : 1a 2d d3 71 8b e6 e1 e7 06 6d f0 f3 a8 7c 5d 72
            54 22 c7 6d 53 d2 6b 2b 43 12 00 37 a0 71 c8 b7             2e fa 19 3d 89 5f 61 58 bb 8c ab b1 cb 22 ca 4a
            dd f3 1d 23 db 68 df 72 f2 90 60 93 84 01 10 58             f2 8a 11 23 3f cc e8 80 09 91 bc 17 7e 4f d5 f8
            97 ba 51 c6 3f eb 6c fd a1 4b c2 7f e0 13 b4 24             f5 ba e2 46 a1 a0 c4 1b 01 a9 94 e0 c2 b8 74 34
            6a 77 63 e6 ce e4 33 8d d1 f4 1f 91 9f 9a 98 2d             e9 de c8 18 78 d7 7d 93 95 0b 67 39 c6 cf 9a 68
            94 e8 62 b5 86 de a3 da 17 bf a7 ea f6 0e 8f 15             a2 32 f6 14 10 6e d0 e3 2f ec fe a4 65 73 a3 70
            fe 16 d7 b3 67 5c ca 4a 4f 75 d8 a9 76 09 55 b8             2a 7b 52 42 d6 b5 f7 64 25 fd 40 16 36 13 ff ad
            5f 03 0f 5d 3e bc cb f8 44 ac e1 61 0d 46 2c b9             82 1d 27 d2 da 69 6c 41 08 9c fc ac f4 8e 8f 3b
            27 f5 70 a8 95 f1 89 80 f7 14 21 04 19 9b 7d 7e             87 0a b7 cd 2c 05 54 c0 03 86 98 e4 d1 47 d8 5e
            9c 29 c0 47 3a 48 cc af 8a b6 4e d4 79 aa ad ed             29 4b c7 2b 50 84 aa 30 4e dd 4d 8d 90 0c ee 4c
            35 34 50 5e 5b f0 be fc 0c 39 96 1a 7b 6f a5 c4             1c 38 d4 56 07 ae b0 5a 83 6b 9d df 79 9e 04 97
            a6 1b e9 ec e5 65 d6 82 3d cd 31 18 2a d3 c1 c9             ed c5 0d 63 3e 53 99 1f 6f 7f 31 f9 75 ef a6 59
            87 c5 3c ee 36 b1 4c 92 42 dc 1e 1c 25 83 e7 4d             92 be 3a dc af c1 33 12 1e bf 66 76 96 b9 44 f1
            56 8c 73 02 a2 2e 64 45 8e d9 74 d0 c3 99 41 ab             db 48 15 bd 9b fb b6 62 6a d9 57 24 c9 20 55 26
            3b 06 32 57 8b e3 05 07 26 40 ff e2 59 b0 9e bd             3c 7a eb e5 45 b4 43 ce 51 b2 5b 35 b3 9f c3 0f
            0a cf 20 0b 7c 30 52 66 2f bb 11 d5 7a 69 5a 6e             a5 85 28 21 49 81 5c 88 77 0e 00 02 a7 37 60 ea
```

Figure 3: S-Box Generated for Key 2

4. PROPOSED AES ALGORITHM WITH KEY MULTIPLICATION

In existing AES algorithm, the 128 bits of state is organized into 4X4 matrix and each byte of state is XORed with byte of round key which is derived by means of key schedule. This block is copied into 4X4 state array and is modified at each stage of encryption and decryption. In modified AES algorithm as proposed in (Mohan H. S. et al, 2011 (a)), by multiplying each byte of state with byte of round key, it is possible to achieve more confusion and more diffusion than key addition which is simple XOR operation. This is due to the fact that the multiplication operation is much complex than addition and also consumes some time. The structure of revised AES algorithm is shown in Figure 4.

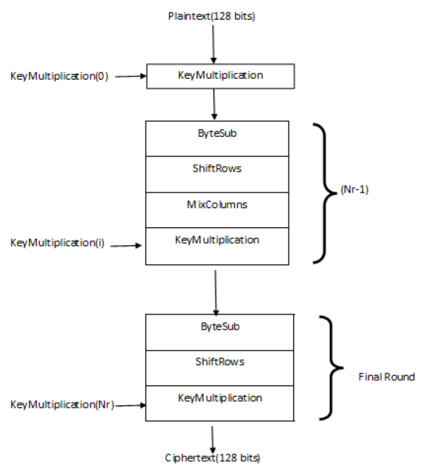


Figure 4: AES with Key Multiplication

4.1. Problem Faced and Solution

The keys can have any value from 0 to 255. If key has value 0, then multiplication of state byte with value 0 will give the byte value 0 which is loss of that state data. To solve this problem, it is checked for value 0 when the key is expanded. If present, then it is replaced by value 1. The inverse of keys is taken by calculating multiplicative inverse of each byte and is used in decryption.

5. PERFORMANCE ANALYSES

Diffusion analysis is made for First order SAC and Higher order SAC considering the following cases:

- Flipping single bit in plaintext while keeping key constant.
- Flipping multiple bits in plaintext while keeping key constant.
- Flipping single bit in key while keeping plaintext constant.

- Flipping multiple bits in key while keeping plaintext constant.

5.1. Time Analysis

The performance of modified AES algorithm is measured in terms of CPU time required for encryption, decryption, key set up, and S-Box generation. These parameters were measured for standard Rijndael algorithm, modified AES with dynamic S-Box only, and modified S-Box with key multiplication only. Corresponding graphs are shown in the figure 5.

Table 1: CPU Time Comparison

	Standard S-Box	Dynamic S-Box	Key Addition	Key Multiplication
Encryption (seconds)	0.20558	0.22388	0.23988	0.22512
Decryption (seconds)	0.28773	0.2683	0.24117	0.33817
Key Schedule (seconds)	0.12726	0.11527	0.14133	0.13263
S-Box Generation (seconds)	1.5555	1.5817	1.5995	1.5052

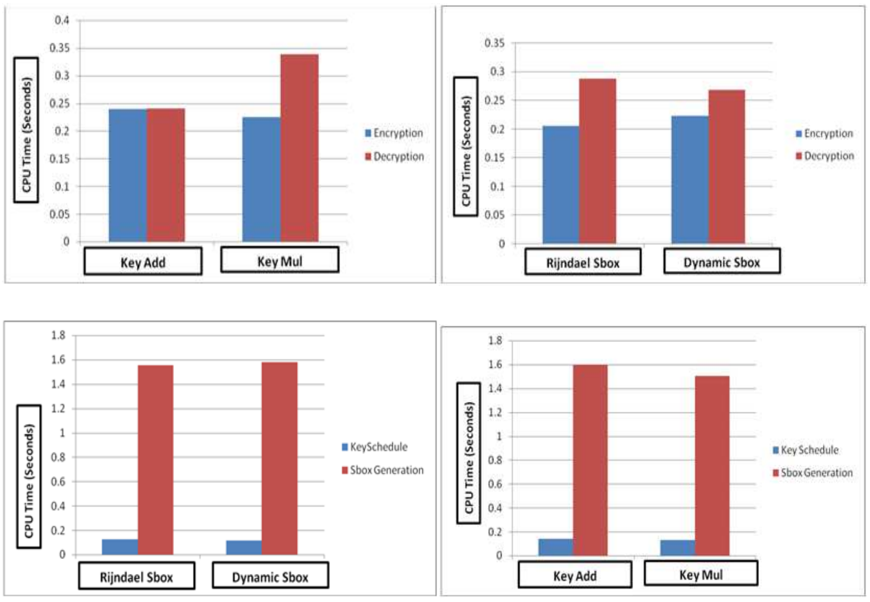


Figure 5: Comparison of Time Required for all Operations with Standard AES and Modified AES

5.2. Diffusion Analysis

The strength of any cryptographic encryption algorithm is estimated by diffusion analysis. The diffusion analysis states how cipher values are sensitive to the input changes made to either plaintext or key. Strict Avalanche Criteria (SAC) is defined to indicate the required diffusion level. In this paper avalanche values were calculated for the four cases which are mentioned in the beginning of this section. First order and higher order SAC values are provided in Table 2.

Table 2: First order and Higher order Avalanche Calculations

	Round Number	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
First Order SAC	Key Addition	20	61	62	71	63	73	67	55	58	58
	Key Multiplication	14	71	68	66	61	72	64	60	60	72
	Standard SBox	12	66	63	61	73	71	59	56	54	55
	Dynamic SBox	17	67	63	66	66	62	79	62	64	67
Higher Order SAC	Key Addition	80	67	76	70	70	56	66	71	56	59
	Key Multiplication	58	70	68	60	62	60	70	63	66	68
	Standard SBox	66	65	71	70	64	63	67	60	64	56
	Dynamic SBox	54	57	64	70	62	61	65	56	70	66

The graphs comparing SAC values for different modules with standard AES are shown in the following Figures. Each graph shows the SAC value after every round. Figure 6 and Figure 7 show the comparison of first order SAC and higher order SAC with key multiplication and dynamic S-Box modules separately.

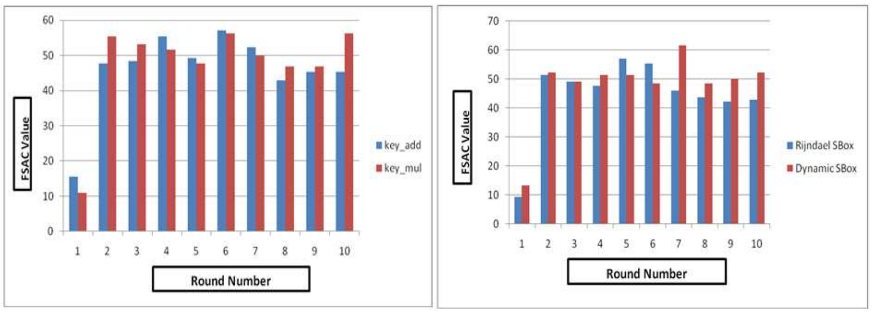


Figure 6: First Order SAC Vs Round Number for Standard AES and Modified AES with Individual Modules

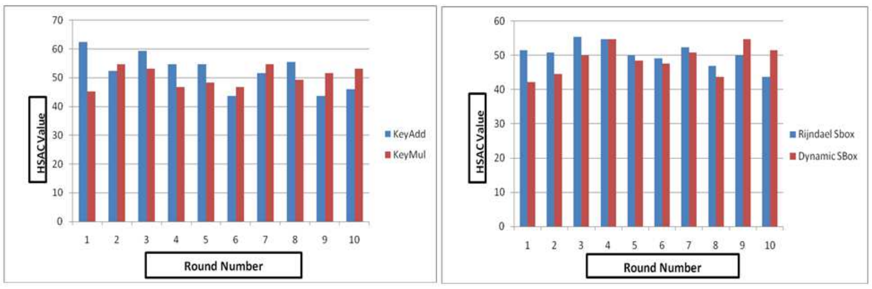


Figure 7: Higher Order SAC Vs Round Number for Standard AES and Modified AES with Individual Modules

The first order SAC was obtained by changing single bit in input bits and cipher value changes for all ten rounds are observed. As it is observed in Figure 6, after tenth round, we have achieved good SAC value with key multiplication and dynamic S-Box. This proved that the diffusion power of modified AES algorithm is more compared to standars Rijndael algorithm. Figure 7 shows the Higher order SAC values. Higher order SAC was obtained by simultaneously changing multiple bits in input bits observing the cipher values for all ten rouds. As it is shown in the graphs, we have achieved good SAC value with modified AES algorithm with key multiplication and dynamic S-Boxes.

Observations were also made for modified AES with both key multiplication and dynamic S-boxes. Table 3 shows the first order and higher order SAC calculations for standard AES and modified AES.

Table 3: First Order and Higher Order Avalanche Calculations

	Round Number	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
First Order SAC	Standard AES	21	62	57	75	70	56	63	64	71	61
	Modified AES	14	66	66	72	72	66	64	58	62	71
Higher Order SAC	Standard AES	68	65	70	56	71	60	59	64	64	61
	Modified AES	63	72	69	72	53	61	58	56	62	66

The graphs comparing SAC values for standard and modified AES algorithm is shown in Figure 8. The graphs show that the modified algorithm with key multiplication and dynamic S-Boxes together yields higher diffusion strength than standard AES algorithm.

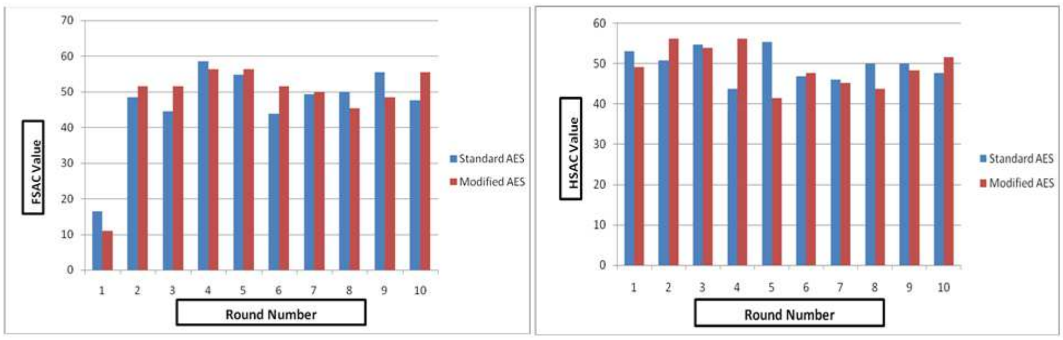


Figure 8: First Order and Higher Order Comparison for Standard AES and Modified AES

6. CONCLUSIONS AND FUTURE ENHANCEMENTS

The basic design of any cryptographic system lies on the diffusion and confusion strength. In this paper we explored the diffusion and confusion elements of AES. Here we also implemented and showed the experimental results of modified AES algorithm with key multiplication and dynamic S-Boxes. Both standard and modified S-Boxes are generated to measure the avalanche value as well as CPU time required for generation. The measured values indicate that the revised AES has higher diffusion and confusion power which increase the security of AES. The higher CPU time required for both of these operations is the indication of higher complexity of the modified algorithm. We can further increase the security of AES by using two keys one for S-Box generation and the other for encryption. The encrypted data can be hidden in image to further protect the data while transferring over Internet. It is possible to make the algorithm further complex and high powered in terms of diffusion and confusion by expanding the state matrix from 4X4 to 8X8 or 16X16 and so on.

7. REFERENCES

1. Mohan H. S and A. Raji Reddy. "Generating the New S-box and Analyzing the Diffusion Strength to Improve the Security of AES Algorithm”, International Journal of Computer and Network Security, Vol. 2, No. 9, September 2010. Chaudhuri, Buddhadeb. (2007). Forest And Tribals A historical Review of Forest Policy, Forest. In Dr.

Chittarajan Kumar Pathy (Ed.), *Forest, Government and the Tribe,* (pp.1-17). New Delhi: Concept of Publishing Company

2. J. Daemen and V. Rijmen, “AES Proposal: Rijndael”, AES Algorithm Submission, September 3, 1999,
3. Abd-ElGhafar, A. Rohiem, A. Diaa and F. Mohammed, “Generation of AES Key Dependent S-Boxes using RC4 Algorithm”, International Conference on Aerospace Sciences & Aviation Technology (ASAT- 13). May 26 – 28, 2009, Military Technical College, Kobry Elkobbah, Cairo, Egypt, 2009.
4. J. Juremi, R. Mahmod, S. Sulaiman and J. Ramli, “Enhancing Advanced Encryption Standard S-Box Generation Based on Round Key”, International Journal of Cyber- Security and Digital Forensics (IJCSDF) 1(3): The Society of Digital Information and Wireless Communications (SDIWC) 2012 (ISSN: 2305-0012), pp. 183-189, 2012.
5. Mohan H. S., A Raji Reddy and Manjunath T. N, “Improving the Diffusion power of AES Rijndael with key multiplication,” International Journal of Computer Applications (0975 – 8887) volume 30– No.5, September 2011(a).
6. Mohan H. S. and A Raji Reddy, “Performance Analysis of AES and MARS Encryption Algorithms,” IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 4, No 1, ISSN Online, 1694-0814, July 2011(b).
7. K. Rahimunnisa, Dr. S. Sureshkumar, K. Rajeshkumar, “Implementation of AES with New S-Box and Performance Analysis with the Modified S-Box”, International Conference on VLSI, Communication & Instrumentation (ICVCI), 2011.
8. Felicisimo V. Wenceslao, Bobby D. Gerardo, Bartolome T. Tanguilig III, “Modified AES Algorithm Using Multiple S-Boxes”, International Conference on Electrical, Electronics, Computer Engineering and their Applications (EECEA2015, 2015.
9. Cryptography and Network Security **-“**William Stallings”, Third Edition.